Um sistema multiagente para o compartilhamento de objetos distribuídos de aprendizagem

Silvana Rossy de Brito[1,2], Maria da Penha de Andrade Abi Harb[1], Eloi Luiz Favero[1], Orivaldo de Lira Tavares[3]

[1]Programa de Pós-Graduação de Engenharia Elétrica (PPGEE),
Universidade Federal do Pára (UFPA)
66075-110 – Belém – PA - Brasil

{srossy,mpenha,favero}@ufpa.br

[2] Núcleo de Tecnologias Interativas de Aprendizagem (NUTEIA),
Instituto de Estudos Superiores da Amazônia (IESAM)
66055-260 - Belém – PA – Brasil

srossy@prof.iesam-pa.edu.br

[3] Centro Tecnológico (CT) - Universidade Federal do Espírito Santo (UFES)
29060-900 - Vitória – ES – Brasil

tavares@inf.ufes.br

Resumo — *Encontram-se descritos, na literatura, diferentes problemas relacionados ao uso de ambientes educacionais baseados em redes de computadores. Dentre as soluções propostas, a adoção de agentes de software tem se destacado na busca por minimizar os problemas enfrentados em atividades cooperativas e colaborativas e ainda por acrescentar, aos ambientes, características que são inerentes aos agentes, como cooperação, iniciativa e autonomia. Uma parte desse esforço tem sido empenhada para promover o compartilhamento e a construção cooperativa de artefatos nos diversos Ambientes Virtuais de Aprendizagem já disponíveis e utilizados atualmente por diversas instituições de ensino no Brasil e no mundo. Este artigo apresenta um Sistema Multiagente para Compartilhamento de Objetos Distribuídos de Aprendizagem (CODAp), no contexto de ambientes educacionais distribuídos, que utiliza o modelo de comunicação P2P, construído com auxílio do framework Java Agent DEvelopment (JADE).*

Palavras Chaves: Modelos de comunicação, comunicação "peer-to-peer", comunicação entre agentes, sistemas multiagente, JADE, AORML.

1. Introdução

Tem sido freqüente, nos últimos anos, a utilização da tecnologia de agentes no projeto e construção de aplicações complexas, devido às características pró-ativas que incorporam em tais aplicações. Esforços na área de redes de computadores, tecnologia de agentes e sistemas distribuídos favorecem, atualmente, o desenvolvimento de espaços virtuais de aprendizagem que apóiem as diversas interações entre estudantes, professores e colaboradores (monitores, especialistas, entre outros). Grande esforço tem sido empenhado para promover o compartilhamento e a construção cooperativa de artefatos nos diversos Ambientes Virtuais de Aprendizagem (AVA's) já disponíveis e utilizados atualmente por diversas instituições de ensino no Brasil e no mundo.

O compartilhamento das construções dos estudantes pode ser feito no contexto de um curso, de uma instituição ou entre diferentes comunidades de aprendizagem. Atualmente, estudantes podem se matricular ao mesmo tempo em vários cursos e as construções e discussões

desenvolvidas no contexto de um curso podem potencializar a aprendizagem de outras comunidades. Essa tem sido a premissa para o desenvolvimento do AmAm - ambiente de aprendizagem multiparadigmático (Harb et al., 2003), baseado em uma arquitetura que considera a existência de comunidades de aprendizagem (e não apenas cursos que se dividem em turmas) e que devem ser organizadas para garantir a cooperação e a utilização de ferramentas diferenciadas entre estudantes. O AmAm é um sistema distribuído que deve facilitar o compartilhamento das produções de estudantes, independentemente das comunidades em que participam. Para facilitar esse compartilhamento, estamos desenvolvendo um sistema multiagentes (SMA) para Compartilhamento de Objetos Distribuídos de Aprendizagem (CODAp). Esse ambiente se insere no contexto de ambientes educacionais distribuídos, é baseado na arquitetura P2P (peer-to-peer) e foi implementado com o *framework Java Agent DEvelopment* (JADE).

Este artigo está organizado da seguinte forma: a seção 2 apresenta os requisitos para um sistema de compartilhamento de Objetos Distribuídos de Aprendizagem, no contexto de ambientes de ensino-aprendizagem, descrevendo as características do sistema, identificação dos agentes, a modelagem segundo uma linguagem extensiva da UML, a AORML; a seção 3 descreve o *framework* JADE para a implementação de SMAs, a seção 4 explora os aspectos de implementação do protótipo desenvolvido, finalmente, na seção 5 são apresentadas as conclusões deste trabalho.

2. CODAp: Um Sistema Multiagente para Compartilhamento de Objetos Distribuídos de Aprendizagem

Para Ferreira e Labidi (1998), a necessidade de incentivar a cooperação entre as pessoas, entre outros fatores, estabeleceu um novo campo dentro da Inteligência Artificial – a Inteligência Artificial Distribuída, que assume a metáfora baseada no comportamento social, onde um grupo distribuído de entidades procura uma solução colaborativa para os problemas. Essas entidades são os agentes e um conjunto de agentes compõe um Sistema Multiagente (SMA). Agentes podem ser organizados em uma sociedade de agentes que interagem, formando uma rede de "solucionadores de problemas" que trabalham juntos para dar soluções a problemas que estão além de suas capacidades individuais, formando, então, um SMA (Brito et al., 2000).

Freqüentemente encontramos na literatura, agentes que povoam os ambientes de aprendizagem cooperativa apoiados por computador (*Computer-Supported Colaborative Learning* – CSCL) para solucionar diferentes problemas que podem ocorrer ao longo do processo educacional. Os ambientes cooperativos CSCL permitem que estudantes trabalhem juntos, compartilhando espaços virtuais e podem ganhar maior flexibilidade e adaptabilidade com a utilização da tecnologia de agentes. Através da tecnologia de agentes, permite-se construir uma sociedade de agentes que trabalhe de maneira cooperativa, considerando agentes humanos externos e agentes artificiais internos no sistema. Nesses ambientes, agentes de software podem atuar como (Santos, 2003): agentes de busca, agentes de filtragem, agentes de monitoração e sinalização, agentes interativos, agentes que lidam com e-mails ou que funcionam como assistentes pessoais.

Em uma comunidade de aprendizagem, os mediadores e os estudantes trabalham juntos para atingir os objetivos educacionais. Segundo Shang et al. (2001), nesses ambientes a colaboração é enfatizada e a competição desestimulada. O papel dos mediadores nesses ambientes sugere a troca de conteúdos, servindo como um colaborador e especialista. O papel do estudante também muda, passando a exercer atividades de construção que devem favorecer a modificação de crenças e valores.

Para apoiar uma comunidade on-line, existem três questões que, segundo Shang et al. (2001), devem ser direcionadas: apoiar vários canais de comunicação incluindo os modelos um-para-um, um-para-muitos, e muitos-para-muitos; achar outras pessoas que compartilham

interesses semelhantes; visualizar e compartilhar contextos comuns. Agentes de software podem apoiar nessas questões através de modelos de interação entre agentes e entre humanos e agentes.

2.1. Características do SMA CODAp

Objetos de aprendizagem são recursos digitais usados em processos de aprendizagem a serem reutilizados em vários contextos. Segundo Beck (2001), citado por David A. Wiley, ***objetos de aprendizagem*** são qualquer recurso digital que possa ser reutilizado para apoio ao processo de ensino aprendizagem. Nesse contexto, os objetos de aprendizagem são considerados **artefatos**, termo que consideramos mais genérico, mas que se apresenta mais apropriado para a ferramenta construída. Artefatos podem ser produzidos por atividades desempenhadas por professores e estudantes, com o uso de ferramentas apropriadas para apoiar essas atividades. Um artefato produzido em uma atividade pode ser insumo por outra atividade, desempenhada por outro aprendiz. A principal idéia dos objetos de aprendizagem é quebrar o conteúdo educacional em pequenos pedaços que possam ser reutilizados em diferentes ambientes de aprendizagem, como na programação orientada a objetos.

O sistema apresentado neste artigo permite a busca e o compartilhamento de artefatos em um ambiente distribuído de aprendizagem e possui como principais características:

- promover suporte ao armazenamento de objetos de aprendizagem para um grupo de estudantes, em diferentes localizações na rede;
- permitir recuperar objetos, independente de sua localização na rede;
- registrar as possíveis localizações de objetos compartilhados;
- registrar a busca de objetos para posterior análise das participações dos estudantes;
- definir automaticamente quais serão as localizações dos estudantes na rede;

O CODAp está sendo desenvolvido incrementalmente, estendendo suas funcionalidades conforme os incrementos são construídos. A primeira fase do processo envolve a construção do sistema para apoiar a interação entre agentes humanos (estudantes) na composição de questionários sobre determinado tema. Um questionário é um artefato composto de perguntas, dicas e respostas (alternativas possíveis). No sistema, um estudante pode elaborar um questionário que pode ser avaliado pelo mediador ou aplicado para outro estudante, através do sistema distribuído. Validada a arquitetura do sistema para essa aplicação, o próximo passo é estender a aplicação para objetos de outras naturezas.

O sistema é modelado segundo a AORML, utilizando-se um diagrama de relacionamentos agentes-objetos. A opção por esta metodologia deve-se as facilidades de utilizar uma abordagem onde o conceito de agente (entidade ativa) não se confunde com as entidades passivas (no caso, os objetos) que serão manipuladas pelos agentes.

2.2 Diagrama de Relacionamento Agentes-Objetos

AORML (Wagner e Tulba, 2003) é uma extensão da UML para modelar sistemas de informação orientados a agentes, com as seguintes características: (1) considera as diferenças ontológicas entre entidades ativas (agentes) e entidades passivas (objetos); (2) considera as organizações e atores de um domínio como agentes (humanos e artificiais); (3) considera os conceitos mentalísticos dos agentes, como compromissos e crenças; (4) captura o comportamento de agentes.

A abordagem de relacionamento objeto-agente (AOR) (Wagner, 2003), na qual a linguagem AORML foi baseada, distingue entre os agentes e objetos de acordo com dois pontos principais: (1) enquanto o estado de um objeto não tem nenhuma estrutura genérica, o estado de um agente tem uma estrutura mentalística, como descrito nas abordagens BDI (Kinny et al., 1996), que consiste em componentes mentais como crenças e desejos; (2) enquanto as mensagens na programação orientada a objetos são codificadas de maneira ad-hoc, uma

mensagem na programação orientada a agentes é codificada como um "ato de fala" de acordo com uma linguagem de comunicação (padrão) de agente, e que deve ser independente da aplicação.

Na AOR, uma entidade é um agente, um evento, uma ação, uma reivindicação, um compromisso, ou um objeto. Os agentes e os objetos são, respectivamente, entidades ativas e passivas, enquanto ações e eventos são as entidades dinâmicas do sistema. Compromissos e reivindicações (cobranças) estabelecem um tipo especial de relação entre agentes.

Somente os agentes podem se comunicar, perceber, agir, fazer compromissos e satisfazer as reivindicações. Objetos são entidades passivas sem tais capacidades. AOR modela agentes artificiais, humanos e agentes institucionais, que são, normalmente compostos de vários agentes humanos ou artificiais. O diagrama de relacionamento agente-objeto para o CODAp é apresentado na figura 1. Esse diagrama é muito similar ao diagrama de classes da UML.

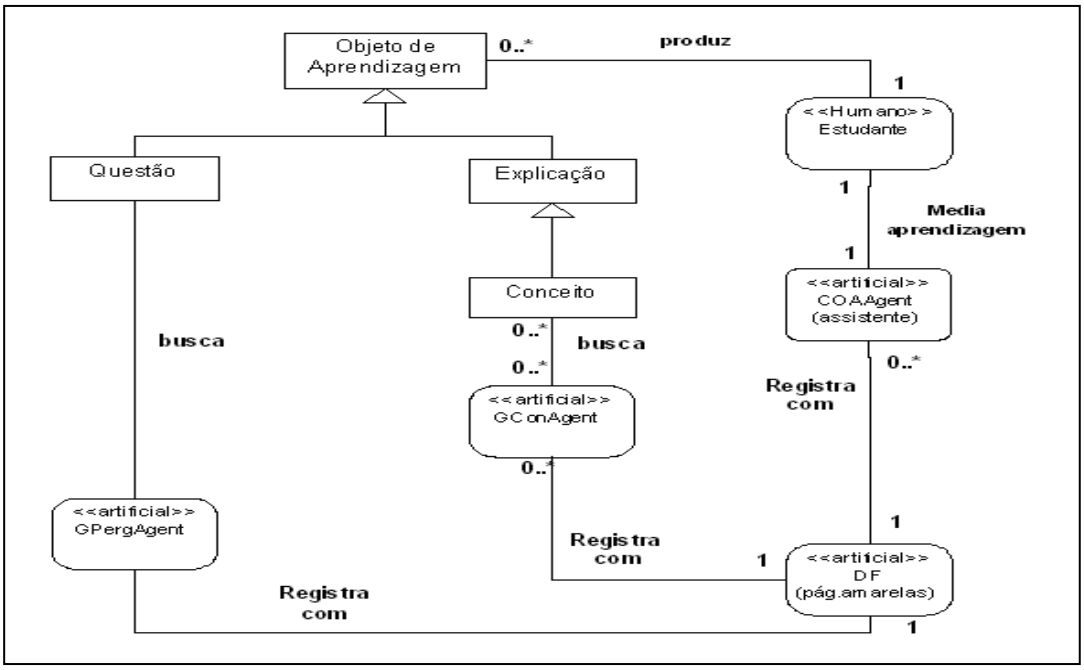

FIGURA 1. Diagrama de relacionamentos agente-objeto do CODAp

2.3. Definição dos agentes

Com base nas características levantadas acima e na delimitação do escopo do problema para a versão preliminar do protótipo gerado, os seguintes tipos de agentes de software foram identificados:

- Agente Consultor de Objetos de Aprendizagem (COAAgent): este agente solicita Objetos de Aprendizagem, neste caso, os objetos de aprendizagem considerados são: **descrição** de um conceito e; **pergunta** associada a um conceito;
- Agente Gerenciador de Conceitos (GConAgent): agente responsável por buscar um Objeto de Aprendizagem do tipo **conceito** solicitado pelo Agente Consultor de Objetos de Aprendizagem;
- Agente Gerenciador de Perguntas (GPergAgent): agente responsável por buscar um Objeto de Aprendizagem do tipo **pergunta** solicitado pelo Agente Consultor de Objetos de Aprendizagem;

Tendo em vista as características do problema e os agentes identificados, a Figura 2 ilustra o processo de execução de uma estratégia pedagógica, incluindo a recuperação e a apresentação dos objetos para o estudante.

Na execução do CODAp (Figura 2), um agente Consultor de Objetos de Aprendizagem (COAAgent) é designado para um estudante e inicia o processo gerando uma estratégia didática para a apresentação dos objetos. Com base na estratégia gerada, o COAAgent solicita o objeto para os Agentes do tipo Gerenciador de Conceitos (GConAgent) ou para Gerenciadores de Perguntas (GPergAgent), conforme a classe do objeto referenciado (se objeto do tipo **pergunta** ou do tipo **conceito**). Os agentes Gerenciadores (GPergAgent e GConAgent) são agentes distribuídos na rede que, após a solicitação de um COAAgent, recuperam e enviam o objeto solicitado. Após o recebimento do objeto solicitado, o COAAgent exibe o objeto para o estudante. Este processo se repete até que todas as ações definidas na estratégia sejam executadas. Uma vez executada uma estratégia para um estudante, o agente COAAgent abandona o SMA, parando sua execução. Os agentes do tipo Gerenciadores permanecem no sistema aguardando solicitações de outros agentes do tipo COAAgent.

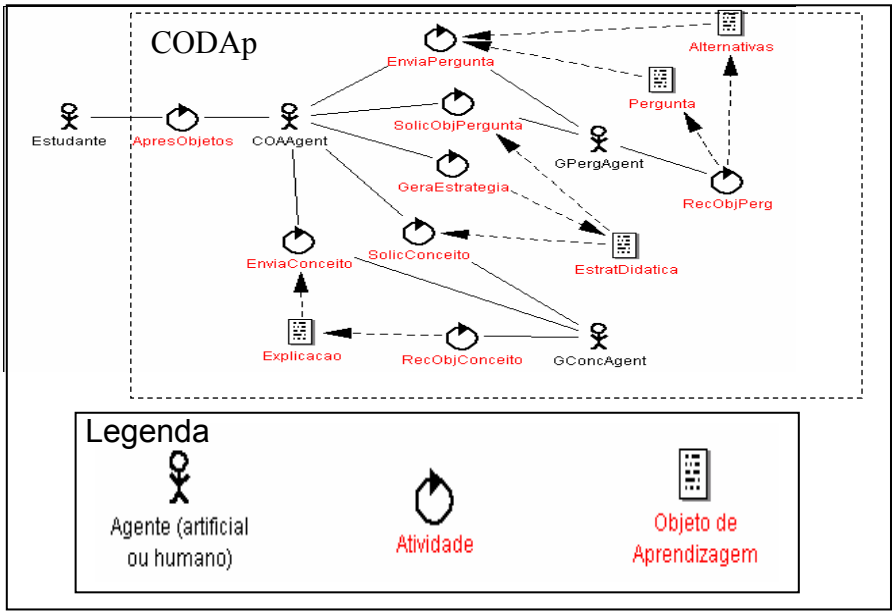

FIGURA 2. Processo de execução do CODAp

2.4. Comunicação entre Agentes

Um aspecto considerado fundamental no desenvolvimento de sistemas baseados em agentes ou SMAs diz respeito a arquitetura de comunicação. Segundo Freitas & Bittencourt (2002), os agentes são vistos, sob o prisma de comunicação em nível de conhecimento, não como servidores, mas como entidades de software capazes de entender e mandar mensagens referentes a determinadas ontologias, e, se projetadas adequadamente, habilitadas à alocação dinâmica de tarefas, repasse (*brokering*), recrutamento, entre outras capacidades, presentes nos atos de fala. Assim, um agente não se confunde com um servidor porque sua interface é mais flexível, dinâmica e semântica.

Recentemente, pesquisadores da tecnologia de agentes têm explorado a arquitetura P2P para implementar a comunicação em SMAs. Os agentes residem nos computadores clientes e informam vários tipos de informação, tanto para a rede, quanto para o cliente. Agentes também podem iniciar tarefas em benefício de outros clientes, podem ser usados para priorizar tarefas em uma rede; mudar o fluxo do tráfego da rede e pesquisar localmente por arquivos. A forma de comunicação "*peer-to-peer*" pode causar até economia no tráfego e no tempo de processamento do sistema como um todo: módulos ou agentes não têm que seguir uma seqüência pré-determinada de mensagens, podendo pular alguns passos de processamento e comunicação de acordo com as mensagens recebidas (Freitas & Bittencourt, 2002). Adicionalmente, a

comunicação como uma ação deliberada permite a implementação da negociação, característica também inerente de agentes em comércio eletrônico, por exemplo.

No modelo P2P, em qualquer momento um agente pode juntar-se à organização e colocar novos serviços à sua disposição (Silva et al., 2002). A tolerância a falhas também é grande, visto que a não continuidade de algum agente não oferece grande impacto ao resto da organização. Obviamente, se a topologia for híbrida, a interrupção dos serviços oferecidos por um agente responsável pelo papel de servidor de meta-dados comprometerá os objetivos de toda ou de parte da organização.

Em uma comparação do P2P com o Cliente-Servidor, a própria finalidade da comunicação é diferente. O conteúdo de uma mensagem pode ser empregado de várias maneiras: para disparar uma inferência ou uma negociação, para reuso (onde o agente passa a dispor de mais conhecimento), ou simplesmente para disponibilizar o conhecimento como documentação para pessoas interessadas no conteúdo das ontologias. Na comunicação Cliente-Servidor, a mensagem pode apenas ser executada no servidor; sua única função é a escolha do objeto e método a ser executado, além da passagem de parâmetros (Freitas & Bittencourt, 2002).

Apesar das vantagens de se implementar organizações de agentes segundo a topologia totalmente descentralizada, uma desvantagem dessa topologia é que, na prática, os algoritmos requeridos para manter uma organização com a comunicação descentralizada são mais complexos dificultando o desempenho de organizações com um grande número de agentes (Silva et al., 2002).

A adoção do um modelo de comunicação centralizado traz inúmeras limitações aos sistemas multiagente: escalabilidade; subutilização de recursos; maior risco de comprometimento do sistema como um todo; necessidade de elevada infra-estrutura na parte do hardware para suportar as aplicações servidoras, sob pena de comprometer o desempenho do sistema. Em contrapartida, modelos de comunicação descentralizados permitem o compartilhamento equilibrado de recursos, usando a capacidade de processamento e armazenamento ociosa nos computadores, além de promoverem a colaboração no processamento de atividades complexas.

3. JADE

O JADE (JADE, 2003) é um *framework* de software para o desenvolvimento de sistemas multiagente, que segue as padronizações da FIPA (*Foundation for Intelligent Physical Agents* – associação sem fins lucrativos que tem o intuito de produzir especificações para interoperabilidade das diversas tecnologias de agente) (FIPA, 2003). O objetivo desse framework é simplificar o desenvolvimento de SMAs ao mesmo tempo em que assegura a utilização do padrão da FIPA. O JADE é, na verdade, um *middleware* que implementa uma plataforma de agentes e um software de desenvolvimento, facilitando o desenvolvimento e o gerenciamento de agentes. A plataforma pode ser distribuída por várias máquinas (que não precisam compartilhar o mesmo sistema operacional) e sua configuração pode ser controlada por uma interface gráfica (GUI) remota.

Na solução do CODAp, utilizamos a plataforma distribuída do JADE, com containeres espalhados entre vários *hosts* sendo que apenas uma máquina virtual Java (*JVM - JAVA Virtual Machine*) é executada em cada *host* e apenas um *host* contém o container principal.

A comunicação usa RMI (*Remote Method Invocation*), porém apenas O JADE Container Agent (container não principal) é um servidor RMI que localmente administra a plataforma.

Jade é completamente implementado na linguagem Java, é *open-source* e possui as seguintes características:

- é escalável (Vitaglione et al., 2002);
- suporta mobilidade de código e estado dos agentes, oferecendo uma interface gráfica de usuário (GUI), para permitir o controle de vários agentes e plataformas ao mesmo tempo (Berger et al., 2003);
- permite implementação de aplicações multi-domínio (a interface simplifica o registro de agentes em um ou mais domínios) (Bellifemine et al., 2003);
- interoperabilidade, já que é compilado com as especificações da FIPA (Bellifemine et al., 2003);
- uniformidade e portabilidade, fornecendo um conjunto homogêneo de APIs que são independentes das características de rede e da versão de Java (Bellifemine et al., 2003);
- é fácil de usar, já que lida de forma transparente ao usuário com todos os aspectos da comunicação entre agentes. Dessa forma, programadores não necessitam conhecer todas as facilidades fornecidas pelo *midleware* (Bellifemine et al., 2003);
- a segurança é preservada já que existem mecanismos para autenticar e verificar os "direitos" dos agentes (Bellifemine et al., 2003);
- fornece gerência do ciclo de vida do agente.

O JADE dispõe, atualmente, das seguintes ferramentas: Agente de Monitoramento Remoto (RMA - *Remote Monitoring Agent*), responsável pela administração e controle da plataforma, sendo que mais de um GUI pode ser ativado (o JADE mantém coerência entre os RMAs através de envio de *multicasting* entre eles); Agente *Dummy,* de monitoramento e Debug para enviar, receber e armazenar mensagens ACL; Agente *Sniffer,* para debugar, "snifar" e salvar em arquivo a comunicação entre agentes, para interceptar as mensagens ACL em trânsito e exibir uma anotação gráfica muito semelhante ao UML que é de grande utilidade para depuração de uma sociedade de agentes ativa; Agente *Introspector*, que permite monitorar o ciclo de vida dos agentes, suas mensagens ACL trocadas e os comportamentos em execução; Agente *SocketProxyAgent,* que age como uma porta bidirecional entre a plataforma e uma conexão TCP/IP; DF GUI, interface de usuário que permite controlar a base de conhecimentos.

4. Implementação do CODAp em JADE

A comunicação entre os agentes é feita através da troca de mensagens em FIPA-ACL. É utilizada a arquitetura P2P, sendo que cada agente se registra no DF do contêiner indicado na inicialização dos agentes (código ilustrado no quadro 1). Os agentes se registram para que possam receber mensagens de outros agentes. A alternativa para receber mensagem sem o registro no DF seria um agente enviar mensagem diretamente para outro, conhecendo, a priori, o seu endereço.

Da mesma forma que os agentes se registram para receber mensagens, o envio de mensagens requer o conhecimento da localização do agente receptor. Para isso, o agente remetente da mensagem necessita buscar, no DF, o endereço de todos os destinatários de mensagens.

O *framework* Jade, dispõe de mensagens de solicitação e mensagens de informação. Para as mensagens de informação foi usado o ato comunicativo "Inform" e para as mensagens de solicitação foi utilizado o "Request".

COAAgent abandona o sistema (doDelete()) quando todos os objetos forem apresentados para o aprendiz e GPergAgent e GConAgent abandonam o sistema quando o tempo limite estabelecido na mensagem de receive bloqueante é alcançado.

No protótipo implementado, as mensagens ACL são utilizadas com dois propósitos: enviar mensagens cujo conteúdo contenha uma string, que é o caso do envio de conceitos, e, enviar mensagens cujos conteúdos contenham objetos, que é o caso da recuperação de perguntas

e alternativas para as respostas. No JADE todas as ACLMessages são codificadas no formato String, definido pela FIPA.

4. 1. Sessão de funcionamento do sistema para um aprendiz

O sistema é iniciado com apenas um agente de cada tipo especificado (COAAgent, GPergAgent, GConAgent), conforme apresentado na interface para gerência remota dos agentes (figura 3). Também é possível iniciar o sistema com mais agentes de cada tipo, bastando, para isso, informar os nomes dos agentes e seus tipos.

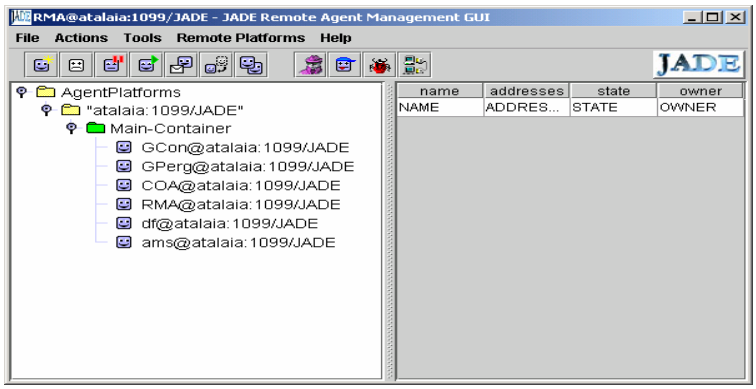

FIGURA 3. Interface Gráfica para gerência dos agentes CODAp

Na execução do CODAp, o agente Consultor de Objetos de Aprendizagem (COAAgent) inicia o processo gerando uma estratégia didática para a apresentação dos objetos. No protótipo implementado, COAAgent gera aleatoriamente uma requisição para os Gerenciadores de Perguntas (GPergAgent). Após o recebimento e a apresentação dos objetos **pergunta** e **alternativas**, COAAgent solicita a descrição do **conceito** associado à pergunta. Todos os agentes podem estar distribuídos na rede, bastando que sejam iniciados no mesmo contêiner. O processo se repete para um número determinado de perguntas, quando, no protótipo, o agente COAAgent abandona o SMA, parando sua execução, conforme se pode observar na Figura 4, que apresenta o *log* de execução do sistema para os três agentes especificados. Os agentes do tipo Gerenciadores permanecem no sistema aguardando solicitações de outros agentes do tipo COAAgent, que podem estar espalhados pela rede. Após um intervalo de tempo especificado, esses agentes também podem abandonar o sistema.

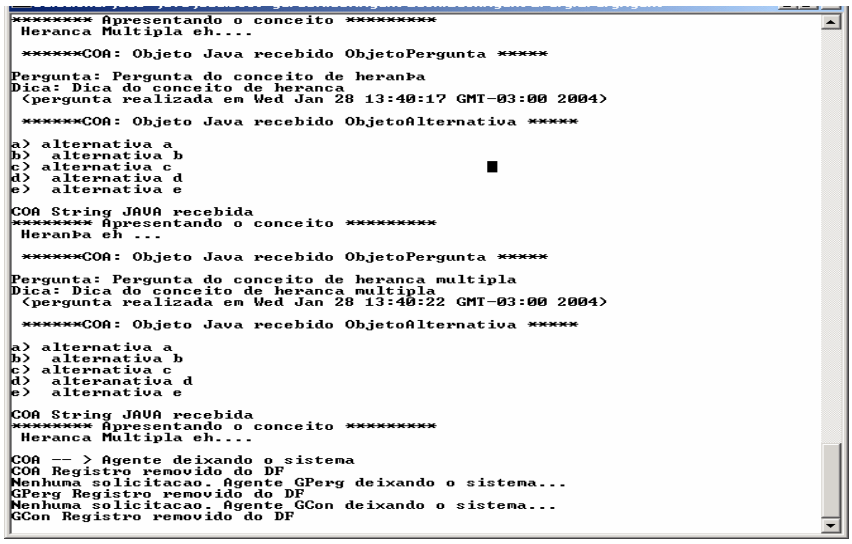

FIGURA 4. Execução apresentando perguntas, alternativas e conceitos.

Através da ferramenta "*sniffer*" é possível interceptar as mensagens ACL, exibir graficamente e salvar em arquivo a comunicação entre agentes (Figura 5). Essa ferramenta é útil para facilitar a compreensão do funcionamento do sistema.

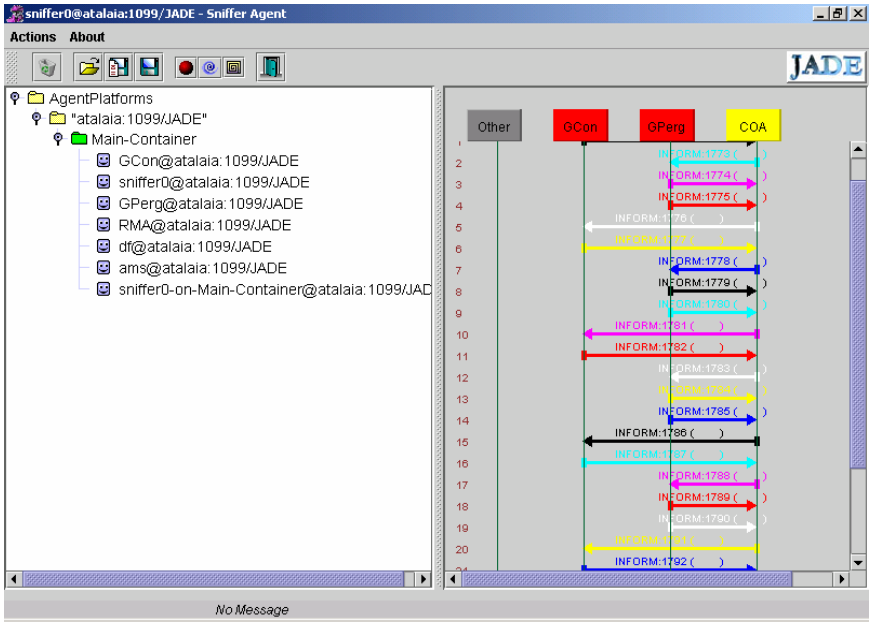

FIGURA 5. Agente *sniffer* mostrando a comunicação entre os agentes no CODAp

A execução em rede, para vários aprendizes é suportada pelo JADE, sendo que em cada máquina é necessário iniciar a plataforma do JADE. As bases de dados podem ser distribuídas e assim, o uso dos recursos é potencializado.

Inicialmente, o alvo de implementação é a criação de um SMA que pode ser facilmente estendido e aperfeiçoado. O aperfeiçoamento do sistema envolve a definição e a recuperação de estratégias didáticas validadas pedagogicamente. Da mesma forma, o SMA produzido não foi executado no contexto de um curso ou ambiente de aprendizagem. Além disso, o conceito de objetos de aprendizagem utilizados aqui se restringiu apenas àqueles suficientes para validar o modelo de comunicação dos agentes, devendo ser extensível para abranger os mais diferentes objetos digitais, inclusive objetos multimídia.

Para efeito pedagógico, o sistema deve evoluir para avaliar as respostas do estudante nas questões de múltipla escolha. É conveniente, para isso, inserir um agente avaliador que receba as respostas do estudante, realizando a avaliação necessária e armazenando um registro de atividades do estudante. Este agente deve ser responsável por interagir com o COAAgente na definição da estratégia didática mais apropriada para aquele estudante.

5. Considerações finais

No contexto das aplicações CSCL, uma gama de aplicações têm sido desenvolvida, utilizando-se a tecnologia de agentes. Utilizando modelos de comunicação flexíveis como o P2P, o processamento concorrente, derivado da característica de distribuição das aplicações, acarreta uma melhora significativa de tempo de resposta e favorecem a tolerância a falhas nessas aplicações. Para a implementação da arquitetura P2P, os agentes do JADE podem fornecer serviços ou consumi-los, o que remove toda a necessidade de distinguir entre clientes e servidores. Assim, é possível que clientes se comuniquem uns com os outros sem a intervenção de um servidor. Além disso, o fato de que a inteligência, a informação e o controle estão distribuídos, permite a construção de aplicações onde a posse é distribuída entre os pares

(agentes) de forma que cada par está hábil, e autorizado a executar apenas um subconjunto das ações da aplicação (Bellifemine et al., 2003).

No contexto das características de agentes, JADE simplifica o desenvolvimento de aplicações que requerem negociação e coordenação entre um conjunto de agentes, onde os recursos e o controle são distribuídos no ambiente. Para a implementação do protótipo apresentado, JADE forneceu bibliotecas e exemplos de fácil compreensão para implementar a comunicação segundo o modelo P2P. Além disso, os agentes em JADE controlam sua própria *thread* de execução, e, portanto, podem ser programados para iniciar a execução de ações sem intervenção humana (garantindo a característica de pró-atividade).

6. Referências

BECK, R.J. **Learning Objects: What?**. Center for Internation Education. University of Winsconsin. Milwaukee. 2001.

BELLIFEMINE, F., CAIRE G., POGGI, A., RIMASSA G. **JADE: A White Paper**. Disponível em http://exp.telecomitalialab.com. Acesso em 02/12/2003.

BERGER , M., RUSITSCHKA S., SCHLICHTE M., D., TOROPOV M. WATZKE. **Porting agents to small mobile devices – the development of the lightweight extensible agent platform**. In: Special Issue on JADE of the TILAB Journal EXP, September, 2003.

BRITO, Silvana Rossy, TOGNERI, Denise Franzotti, TAVARES, Orivaldo de Lira, MENEZES, Crediné Silva, FALBO, Ricardo de Almeida. **Um Sistema Multiagente para Gerência de Reuniões em Ambientes de Aprendizagem Cooperativa**. In: Workshop de Ambientes de Aprendizagem baseados em Agentes, 2.,2000, Maceió: UFAL, nov. 2000.

FERREIRA, Jeane; LABIDI, Sofiane. **Modelagem do Aprendiz baseado no Paradigma de Ensino Cooperativo**. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 11, 1998, Fortaleza. **Anais…** Fortaleza: Universidade Federal do Ceará, 1998.

FIPA. **Foundation for Intelligent Physical Agents**. http://www.fipa.org/. Acesso em 20/11/2003.

FREITAS & BITTENCOURT, 2002 . **Comunicação entre Agentes em Ambientes Distribuídos Abertos: o Modelo "peer-to-peer"** Disponível em: http://www.sbc.org.br/reic/edicoes/2002e2/informativos/ComunicacaoEntreAgentesEmAmbientesDistribuidosAbertos-OModeloPeerToPeer.pdf. Acesso em 10/11/2003.

HARB, M. P. A. A., BRITO, S. R., SILVA, A. S., FAVERO, E. L., TAVARES, O. L., FRANCÊS, C. R. L. **AmAm: ambiente de aprendizagem multiparadigmático**. multiparadigmático. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 2003, Rio de Janeiro. Rio de Janeiro: NCE-IM-UFRJ. p. 223-232.

JADE. **Java Agent DEvelopment Framework**. Disponível em: http://sharon.cselt.it/projects/jade/. Acesso em: 25/11/2003.

KINNY, D., GEORGEFF M., RAO, A. **A methodology and modelling technique for systems of BDI agents**. In: W. van der Velde, J. Perram (eds.): Agents Breaking Away. Proceedings on Workshop on Modelling Autonomous Agents in a MultiAgent World, MAAMAW-96, 7, pag. 56-71. Springer-Verlag: Berlin, Germany, 1996. MAAMAW'96, LNAI 1038, Springer, Berlin,Germany,1996.

SANTOS, Neide. **Agentes de Software em Ambientes Educacionais Mediados por Computador.** Revista Brasileira de Informática na Educação. Porto Alegre - v.11, n.1, p.9 - 25, 2003.

SHANG, YI; SHI, HONGCHI; CHEN, SU-SHING. An intelligent distributed environment for active learning. In: **ACM Journal of Educational Resources in Computing (JERIC)**. Vol. 1, No. 2, Summer 2001, Article 4, 17 pages. ISSN:1531-4278. New York: ACM Press.

SILVA, A S., HERNÁNDEZ-DOMÍNGUEZ, A., FAVERO, E. L., BRITO, S. R., TAVARES, O. L., MENEZES, C. S. (2002) **Organizações de agentes em ambientes de aprendizagem**, In: Workshop de Ambientes de Aprendizagem baseados em Agentes, 4,2002, São Leopoldo: UNISINOS, 2002.

VITAGLIONE, G., QUARTA, F.., CORTESE, E. **Scalability and Performance of JADE Message Transport System.** In: AAMAS Workshop on AgentCities, Bologna, 16th July, 2002.

WAGNER , G. **The Agent-Object-Relationship Meta-Model: Towards a Unified View of State and Behavior.** In: Information Systems 28:5 (2003), pp. 475-504.

WAGNER , G., TULBA, F. **Agent-Oriented Modeling and Agent-Based Simulation.** To appear in P. Giorgini and B. Henderson-Sellers (Eds.), Proceedings of 5th Int. Workshop on Agent-Oriented Information Systems (AOIS-2003), ER 2003 Workshops, Springer-Verlag, LNCS, 2003.